Recurrent Neural Networks for Weather Forecasting using LSTM

Nishanth Badim
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas, Richardson, Texas, US
nxb220044@utdallas.edu

Harika Gattu
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas, Richardson, Texas, US
hxg220030@utdallas.edu

***Abstract*—Time series prediction can play an important role in a variety of contexts, such as weather forecasting, river flow forecasting, and stock price forecasting, and they may aid individuals in making wise decisions. The ability to anticipate the weather is essential in many industries, including disaster management, transportation, and agriculture. The hidden layer of a recurrent neural network's LSTM (Long Short-Term Memory) is an iterative structure that can capture time series' long-term dependencies. This study explains how to use the Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) technique to predict the temperature for the following day based on the temperature for the previous day using historical data. Due to its distinctive storage structure, the Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) removes long-term reliance, which helps with weather time series prediction. We also train the model using a variety of parameters. The accuracy of the calculation is determined by Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Squared Error(MSE), and R-squared Score(R2 Score). In order to gain a better understanding of the data utilizing the training set and test set, the results are graphed.**

***Keywords*—RNN, LSTM, Weather, Time series, Temperature, Prediction, Hidden Layers, Metrics**

I. INTRODUCTION

As atmospheric conditions are so complicated and variable, predicting the weather is a challenging task. For example, arranging transportation, planning agricultural activities, and preparing for extreme weather events, accurate temperature forecast is essential. By examining the initial values of pertinent atmospheric features that are gathered by meteorological measurements, weather forecasting is the process of predicting the future state of the atmospheric conditions of a certain location. In time series forecasting, the development of deep learning algorithms, particularly RNN and LSTM, has produced encouraging results.

For processing sequential data with temporal dependencies, LSTM and RNN have emerged as prominent resilient neural network designs in recent years. They are well-suited for simulating the intricate and dynamic character of weather patterns because they can detect long-range patterns in time series data. These models are remarkably accurate in predicting future weather conditions by taking past context for meteorological factors into account. In this study, we employ RNN and LSTM to forecast the temperature based on the current temperature.

The Recurrent Neural Network is best suited for processing sequential data due to its large short-term memory design, which enables it to remember input for an extended period of time. You will need to adjust the weights once you've finished studying. Recurrent neural networks draw on earlier stages to learn data and predict future trends. The Long Short-Term Memory is where the data is kept before being used by the Recurrent Neural Network. It is now feasible to foresee and operate dynamically.

In addition to multi-dimensional data sets, one-dimensional data sets can also be utilized to forecast the weather. For one-dimensional data sets, temperature estimation is often conducted using only the temperature variable. In multidimensional data set problems, in addition to temperature information, weather forecasts can be created by using values such as rain, pressure, wind speed, wind direction, and humidity of the relevant place.

II. BACKGROUND WORK

Effective weather prediction models must be built through the use of feature engineering and data preprocessing. The performance of the model might be negatively impacted by the existence of missing data and outliers, which are frequent in weather datasets. Because of this, using reliable data imputation methods, such as the Imputer included in the code described here, guarantees the dataset's consistency and completeness.

Feature scaling is used to improve the model's capacity to comprehend the data. The "tmax" values in this code are transformed into a constant range between 0 and 1 by normalizing them with MinMaxScaler. Scaling the features enables a more evenly distributed convergence of optimization techniques and prevents dominating factors from overpowering the model during training.

Prior to scaling, the features are organized with the help of the VectorAssembler, which produces a single temperature vector that is used as input to the MinMaxScaler. This data preparation makes it simpler for the RNN and LSTM models to process the input during the training process.

All things considered, combining RNN and LSTM models with data preprocessing methods like imputation and feature scaling has considerable promise for enhancing the precision of weather forecast predictions. It is anticipated that the application of these techniques will result in more dependable and sturdy models, which can greatly improve weather forecasting systems. In this study, we provide a detailed examination of the methodology suggested, demonstrating its efficiency through thorough tests and analyses on historical weather data.

III. THEORETICAL AND CONCEPTUAL STUDY

A. Time Series Prediction in Weather

In the context of weather, a time series is a collection of observations that were made over time and were often taken at regular intervals. Weather datasets frequently contain missing values, outliers, and noise as a result of a variety of problems, including sensor failures or mistakes during data collection. For addressing these problems and maintaining data quality for precise analysis, preprocessing techniques including data imputation, outlier detection, and smoothing are essential.

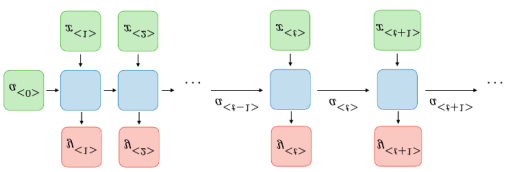
B. Recurrent Neural Networks (RNN)

A standard RNN Recurrent neural networks, or RNNs, are a subclass of neural networks that allows the use of hidden states and previous outputs as inputs.

The hidden state of the recurrent network cascades forward to affect the processing of each incoming example, traversing several time steps while keeping the sequential data. Finding relationships between events that occurred over an extended period of time is known as "long-term dependencies" because a subsequent event depends on and is caused by one or more earlier occurrences.

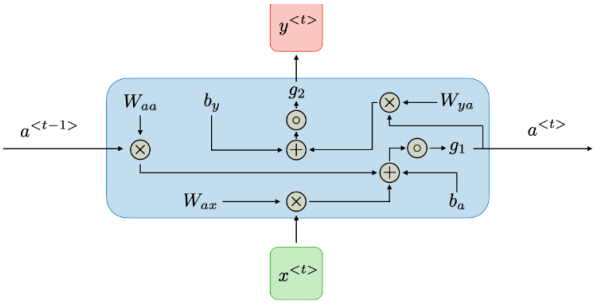
$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right)$$

As an example, H_t represents the hidden state at time step t, which is determined by a combination of the input at the same time step x_t and the prior hidden state h_t-1, both of which have been modified by weight matrices W and U, respectively. These weight matrices serve as filters, defining the relative importance of the present input and the previous concealed state. Any errors produced by these weight matrices will be transmitted back and used to change their values until the error cannot be lowered any further..



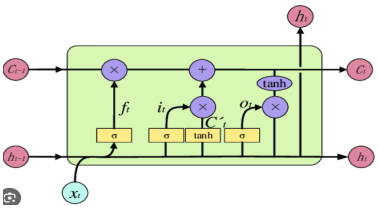
The function, a common tool for weight input and hidden state squashing, is either tanh or a logistic sigmoid function, depending.

Standard recurrent neural networks (RNNs) have difficulties with short-term memory because of a vanishing gradient problem that occurs when working with longer data sequences. In the context of RNNs, vanishing and exploding gradient phenomena frequently occur. They occur because multiplicative gradient, which can exponentially decrease or increase with respect to the number of layers, makes it impossible to capture long-term interdependence.



C. Long Short term memory architecture (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically is made to deal with sequential data, such time series, audio, and text. Language translation, speech recognition, and time series forecasting are among the tasks that LSTM networks are particularly suited for because they can learn long-term dependencies in sequential data. Because the model's repeating module consists of a combination of four layers that interact with one another. The LSTM architecture is a solution to the vanishing gradient problem encountered in traditional RNNs.



A LSTM network consists of the following components.

Forget Gate:

The primary characteristics of an LSTM are its ability to remember and recognize information entering the network and to reject information that is not necessary for the network to learn from the input and make predictions. This LSTM feature is brought about by this gate. It aids in determining if data can transit through the network's layers. The information from the preceding levels and the information from the current layer are the two different forms of input.

Input Gate :

The input gate is a vital component that regulates the information flow into the cell state. It helps in determining the relevance or significance of new input data in the context of the previous information stored in the cell state.

Output gate :

The output gate assists in determining the network's subsequent hidden state, which is where data passes via the sigmoid function. The updated cell from the cell state is sent through the tanh function and multiplied by the output state's sigmoid function. It facilitates information transmission in the hidden state.

Cell state:

This layer estimates the cell state, through which information with regard to weight growth goes. In the cell state, the results of the input gate and forget gate are multiplied by one another. Values very close to zero maximize the amount of potentially lost information.

D. Root Mean Square Error

The average magnitude of a predictive model's errors is typically measured using the metric known as RMSE. It computes the square root of the squared average difference between the expected and actual values. In situations when major errors are more important, RMSE is especially helpful because it condemns large errors more harshly than tiny ones.

The RMSE statistic is used in weather forecasting to evaluate how accurately the model predicts periodic meteorological factors like temperature or humidity. Lower RMSE values signify better accuracy in identifying the data's broad trends and patterns.

E. R2 Score (R-squared Score)

R2 Score, quantifies the percentage of variance in the dependent variable (the goal variable) that can be predicted from the independent variables (the projected values). It shows how well the model fits the data and gives information about how effectively the model accounts for variation in the data.

R2 Score is modified in time series analysis to take the temporal aspect of the data into consideration. A naive forecast, such as employing the previous value of the variable of interest as the prediction, would account for the majority of the variation explained by the model, according to this adjusted R2 Score.

When predicting the weather, the R2 Score is used to assess how well the model accounts for the variation in tomorrow's Tmax(maximum temperature) based on today's Tmax(maximum temperature). A higher R2 Score indicates that the model is more capable of identifying and utilizing temporal patterns to provide more accurate predictions.

F. Mean Absolute Error

The average absolute difference (MAE) between the values that were predicted and the actual values is what is measured. No matter which way the forecasts are going, it gives a clear evaluation of how far they are from the actual values.

Understanding MAE's average prediction error for Tmax is helpful when making weather predictions. In contrast to RMSE, it provides a more easily understood measure of accuracy because it takes into account the absolute deviations rather than the squared differences.

G. Mean Squared Error

MSE determines the average of the squared differences between anticipated and actual values, much like RMSE but without taking the square root. When the model needs to take into consideration outliers and extreme values, it typically accentuates greater errors.

MSE can be used to evaluate a model's resilience in the context of weather forecasting by determining how well it tackles anomalies and unusual weather occurrences.

IV. METHODOLOGY

A. Data Preprocessing

The dataset used here is weather data that is obtained from Kaggle. This dataset consists of 13,509 instances and five features recorded on a particular day from 1970 to 2022. The features include maximum temperature(F), minimum temperature(F), rainfall (cm) recorded, and the maximum temperature(F) of the next day.

Then it is normalized in the range 0 to 1 using the MinMax scaler. The purpose of normalization is to convert the values of the dataset's numerical columns to a standard scale, without distorting or losing distinctions in the value ranges information.

B. RNN Model using LSTM

The dataset is categorized into 3 sets:

- 80% of the original dataset is used as training data, which is then adjusted before being fed into the RNN model.
- 20% of the original dataset was used for testing, and it was reshaped appropriately for the RNN model.
- 20% of the training set served as validation data.

The dataset is categorized and prepared according to the above parameters and it is fed into the LSTM based RNN model.

Model Architecture:

We define the architecture of the Long Short-Term Memory (LSTM) model. initializes the parameters (weights and biases) for the LSTM model. The parameters include weight matrices (w_f, w_i, w_g, w_o, w_v) and biases (b_f, b_i, b_g, b_o, b_v) for the forget gate, input gate, candidate gate, output gate, and the relation between the hidden state and the output. The weight matrices are randomly initialized with values from a standard normal distribution, and the biases are set to zero vectors. These parameters are essential for the LSTM model's operations during the forward and backward passes, enabling it to process input data and learn from training examples.

Activation Functions:

The sigmoid activation function was utilized for the input, forget, and output gates of the LSTM model. This function is defined as follows:

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

where x represents the weighted sum of inputs and biases for the respective gates.

The tanh activation function was applied to the cell state and output of the LSTM model. The tanh function is defined as:

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

where x denotes the weighted sum of inputs and biases for the cell state and output computation.

Forward pass:

The LSTM parameters include weight matrices and biases for the forget, input, candidate, and output gates, as well as weights and biases for the output layer. It computes the forward pass computations for each time step, and stores them in a separate lists every time by iterating over the input data. It calculates the forget gate, input gate, candidate, memory state, output gate, hidden state and logits output at each time step.

During the forward pass, the LSTM model calculated the activations of the sigmoid function for the input, forget, and output gates, as well as the tanh activation for the cell state and output. These activations were used to control the flow of information and to determine the relevant information to store or discard at each time step.

Backward pass:

The gradients of the model's parameters with respect to the loss function are calculated during the backward pass. This is an essential stage in training the LSTM model since it enables us to use gradient descent to update the parameters and enhance the model's performance on the training set of data. The mean squared error (MSE) loss between the outputs of the model and the target sequences at each time step is calculated during the backward pass.

The gradients of each gate and cell state at each time step are then determined using backpropagation through time (BPTT). The gradients can optionally undergo L2 regularization to avoid overfitting. To prevent gradients from bursting during training, the gradients are finally clipped using gradient norm clipping. The function returns the total loss and the cumulative loss.

Training, Validation and Testing:

We train the LSTM model using a training set and monitor its performance on a separate validation set during training. The model's training loss and validation loss are tracked over multiple epochs. After training, the model's final evaluation is performed on a test dataset to assess its performance on new, unseen data.

Hyperparameters, such as the number of epochs, hidden state size, learning rate, window size and input feature size, are set before training. The training loop updates the model's parameters using forward and backward passes to minimize the loss.

As both the training loss and the validation loss (Fig 5.3) show a similar downward trend during training, indicating that the model is learning useful patterns from the training data and generalizing effectively to new data.

Evaluation:

We evaluate the LSTM model's performance on both the training and test sets. During this evaluation, we execute the forward pass for each input sequence, generating the corresponding output sequence and predictions. For testing set, we evaluate the LSTM model's generalization capability to previously unseen data. The test set is comprised of samples that the model has not encountered during training, making it an important benchmark for real-world scenarios. To quantify the model's accuracy, we calculate essential performance metrics, including the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R2) score on the training data and testing data.

In conclusion, these indicators are crucial tools for evaluating how well weather prediction algorithms are working. When considered collectively, they offer a thorough assessment of a model's accuracy, resilience, and predictive powers, enabling researchers and forecasters to decide which models are most appropriate for precise weather forecasting.

The results of both evaluations are showcased in the Results section, providing valuable insights into the LSTM model's learning capabilities and its ability to generalize to unseen data.

V. RESULTS AND ANALYSIS

Using the proper error/loss function, activation functions, number of hidden neurons, window size, and number of epochs can have a major impact on the model's performance and help provide more accurate weather predictions when used in conjunction with RNN and LSTM models for forecasting the weather. Building efficient and precise prediction models requires taking these factors into account.

A. Error or Loss functions

When a predictive model is being trained, the error or loss functions track the difference between the predicted and actual values. Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) are popular error functions in the context of time series prediction for weather forecasting. By reducing error during training, these functions aid in quantifying the precision of the model's outputs and directing the optimization procedure.

B. No. of hidden layer size

The neural network's ability to recognize and grasp patterns in data is determined by the number of hidden neurons contained within its hidden layers. Choosing the right number of hidden neurons is critical for time series prediction because it prevents the data from being overfitted or underfitted. The number of hidden neurons can be fine-tuned using techniques like cross-validation to get optimal prediction performance.

C. Window size value

The term "window size" refers to the amount of prior time steps (lags) taken into account as input features to forecast the following time step. In order to train the model for time series prediction, input-output pairs must be created with the proper window size. Long-term dependencies might be captured by a bigger window size, but this could result in more complex computations and possible data leaking. But a window of lesser size might not be able to capture enough information to make reliable predictions.

D. No. of epochs considered

When a neural network is being trained, the number of epochs indicates how many times the full training dataset is run through the network. To strike the correct balance between overfitting and model convergence, this hyperparameter must be carefully controlled. A model may be underfit if there are insufficient epochs, while an overfit model may develop from memorizing the training set. When the model performance on the validation set begins to deteriorate, early stopping approaches can be used to interrupt the training process and prevent overfitting.

We have conducted an experiment taking combinations of parameters like No.of input layers, No.of output layers, No.of hidden neurons, No.of epochs, Learning rate, Window size, and analyzed the model's performance, and the results are presented. The following graphs illustrate the outcomes of the experiments. Each graph focuses on data from the Weather forecasting train and test sets, showcasing temperature values plotted against time.

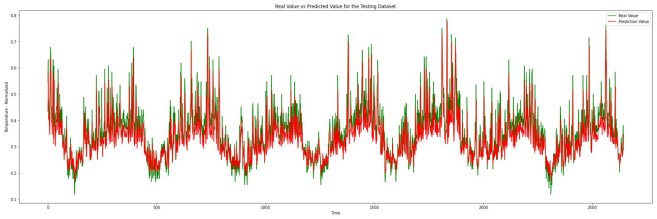


Fig 5.1: Real Value vs Predicted Value for the Testing Dataset

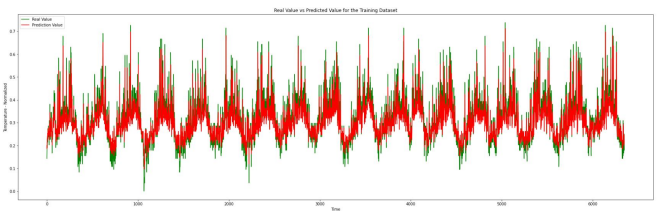


Fig 5.2: Real Value vs Predicted Value for Training Dataset

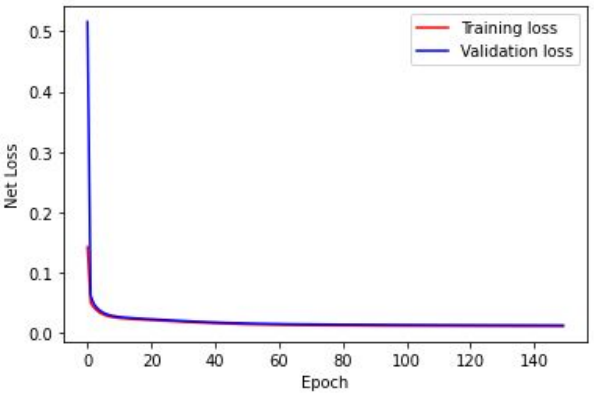


Fig 5.3: Training loss vs Validation Loss

We evaluated our model using multiple parameter combinations, and the model produced the outcomes shown below.

Index	Parameters	Results
1	No of input layers: 1 No of output layers: 1 No of hidden neurons: 25 No of epochs: 30 Learning rate: 1e-14 Window Size:5	RMSE for training set: 0.06500876147998044 RMSE for testing set: 0.06885106377725153 MAE for training set: 0.04911514348661712 MAE for testing set: 0.052857901456730215 R2 Score for training set: 0.5632640118054114 R2 Score for testing set: 0.5157383137966081
2	No of input layers: 1 No of output layers: 1 No of hidden neurons: 25 No of epochs: 25 Learning rate: 1e-16 Window Size:5	RMSE for training set: 0.060748086896024484 RMSE for testing set: 0.06440905753009336 MAE for training set: 0.04597007600063794 MAE for testing set: 0.04833433496621848 R2 Score for training set: 0.618772959394635 R2 Score for testing set: 0.5776853080473536
3	No of input layers: 1 No of output layers: 1 No of hidden neurons: 40 No of epochs: 90 Learning rate: 1e-2 Window Size:5	RMSE for training set:0.059497855404445514 RMSE for testing set: 0.06305131250338868 MAE for training set: 0.04391897304878684 MAE for testing set: 0.047177061979234446 R2 Score for training set: 0.6343032417165546 R2 Score for testing set: 0.5953024598812233
4	No of input layers: 1 No of output layers: 1 No of hidden neurons:20 No of epochs:100 Learning rate:1e-3*6 Window Size:5	RMSE for training set: 0.05909957187422058 RMSE for testing set: 0.06319659458870379 MAE for training set: 0.0442367600213438 MAE for testing set: 0.046735739415908466 R2 Score for training set: 0.6391828628400491 R2 Score for testing set: 0.5934353126980128
5	No of input layers: 1 No of output layers: 1 No of hidden neurons: 25 No of epochs: 40 Learning rate: 1e-12 Window Size:5	RMSE for training set:0.05721865508614595 RMSE for testing set: 0.06274322365154712 MAE for training set: 0.04169220070533249 MAE for testing set: 0.04525769780322782 R2 Score for training set: 0.6616622262646095 R2 Score for testing set: 0.5978459402969563
6	No of input layers: 1 No of output layers: 1 No of hidden neurons:15 No of epochs:150 Learning rate:1e-3 Window Size:3	RMSE for training set: 0.055678186077213886 RMSE for testing set: 0.05752448292694825 MAE for training set: 0.041935737114744055 MAE for testing set: 0.04243199267363312 R2 Score for training set: 0.6798795176687309 R2 Score for testing set: 0.6631413151087961
7	No of input layers: 1 No of output layers: 1 No of hidden neurons:20 No of epochs:150 Learning rate:1e-3 Window Size:3	RMSE for training set: 0.05494593535057079 RMSE for testing set: 0.05704845295479308 MAE for training set: 0.04121696396476582 MAE for testing set: 0.04184231752195152 R2 Score for training set: 0.6882442674128342 R2 Score for testing set: 0.6686934324685588

VI. CONCLUSION AND FUTURE WORK

Finally, it should be noted that time series prediction and analysis for weather forecasting are important and difficult jobs that call for advanced methodologies to capture the intrinsic temporal patterns and complexity of weather data. In this theoretical and conceptual study, we looked into many different areas of this subject, such as evaluation metrics, model architectures, data preprocessing, and hyperparameter tweaking.

We talked about crucial assessment metrics, like RMSE, R2 Score, MAE, Mean Squared Error, and R-squared Score, which are frequently used to rate the effectiveness of weather prediction models. Researchers and meteorologists can use

these measures to make wise choices regarding the model selection and improvement since they offer insightful information about the models' precision, robustness, and generalization abilities.

In-depth examination of important time series prediction model elements, including error/loss functions, activation functions like sigmoid and tanh, the number of hidden neurons, window size, and epochs, was also conducted. These elements are essential to the modeling process because they affect how well the model can capture temporal relationships, manage non-linear data, and avoid overfitting.

There are many untapped research possibilities in the field of time series prediction in weather forecasting, which is still under development. The following are some prospective study directions:

- Investigate the use of more sophisticated deep learning architectures, like Transformers or attention-based models, in the forecasting of meteorological time data. These structures have demonstrated promise in a number of sequence-to-sequence applications and may enhance weather data modeling capabilities.
- Examine the efficacy of ensemble approaches for forecasting the weather, which combine several models to improve prediction accuracy and robustness. In order to address the shortcomings of individual models and produce more accurate forecasts, ensemble approaches, such as stacking or bagging, can be used.
- Work on creating interpretable and explicable weather prediction models so that people may comprehend the justifications for the predictions and acquire confidence in them.

In conclusion, interdisciplinary efforts are needed to progress the field of time series prediction for weather forecasting, encompassing developments in machine learning, deep learning, meteorology, and climate research. The results of such study could result in more precise and trustworthy weather forecasts, which would be advantageous to many industries and would improve methods for managing resources, preparing for disasters, and adapting to climate change.

VII.REFERENCES

[1] Dataset - https://www.kaggle.com/datasets/awsaf49/clean-weather-dataset

[2] A Complete Guide to LSTM Architecture and its Use in Text Classification - https://analyticsindiamag.com/a-complete-guide-to-lstm-architecture
Vanishing gradient and exploding gradient in Neural networks

[3] Vanishing gradient and exploding gradient in Neural networks - https://arunm8489.medium.com/vanishing-gradient-and-exploding-gradient-in-neural-networks-15950664447e

[4] Recurrent Neural Networks cheatsheet - https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[5] Recurrent Neural Networks: An Embedded Computing Perspective-https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9044359

[6] Weather Prediction using Long Short Term Memory (LSTM) model - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10061039

[7] A Recurrent Neural Network Model for Weather Forecasting -https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8907196

[8] Machine Learning Based Weather Prediction Model for Short Term Weather Prediction in Sri Lanka - https://www.researchgate.net/publication/353931832_Machine_Learning_Based_Weather_Prediction_Model_for_Short_Term_Weather_Prediction_in_Sri_Lanka

[9] Long Short Term Memory (LSTM) and how to implement LSTM using Python - https://waliamrinal.medium.com/long-short-term-memory-lstm-and-how-to-implement-lstm-using-python-7554e4a1776d

[10] https://github.com/VikParuchuri/zero_to_gpt/blob/master/explanations/rnn.ipynb